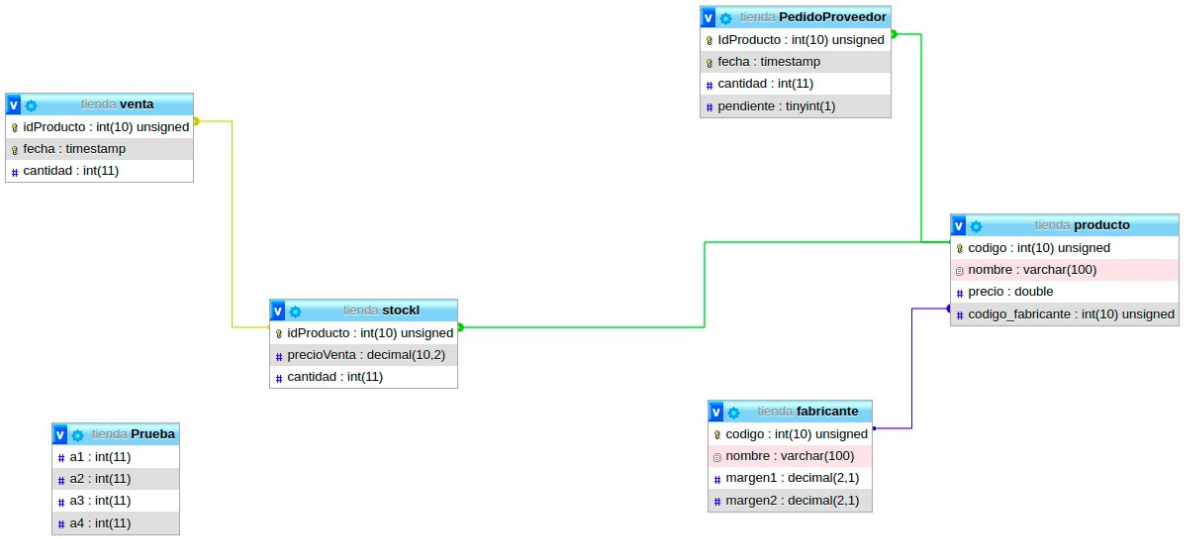
Partiendo de la BBDD tienda realiza los siguientes apartados. Al terminar el examen se os quedará un modelo relacional como el siguiente



1. Cambia la t**abla producto**s, añadiendo un nuevo campo “precio” DECIMAL(10,2), esta columna contendrá el precio de compra del producto en la tienda, (la columna precio hace referencia al precio de compra del producto).

   Nota: El precio de la tabla producto es el precio de compra de dicho producto. Es decir lo que le cuesta a la tienda comprar ese producto.

   0,25 ptos

2. Crea una **tabla llamada stock**, dicha tabla contiene los campos
   - idProducto tipo int int(10) UNSIGNED, autoincremental, foránea respecto a la tabla Producto. Si se borrara un producto de la tabla Producto desaparecerá todo registro de esta tabla que haga referencia a él.
   - precioVenta tipo int(10,2). Ten en cuenta que el precio de venta puede ser null,
   - cantidad tipo int, no puede ser null y por defecto es 0, además nunca podrá ser negativo.

   Modifica la tabla fabricante añadiendo dos columnas Margen1 y Margen2 que son del tipo decimal(4,2) y son los porcentajes que deja el fabricante aplicar a sus productos.

   Nota: Estos campos podrán tomar el valor 0.1 significa un 10%, 0.01 significa un 1%...
   1 ptos

3. Crea una **función “PrecioVenta”** que recibe como parámetro el idProducto int(10) y retorna el precio de venta :decimal(10,2) teniendo en cuenta:
   Si el precio es menor o igual a 100 euros el precio se calcula con el precio de venta del producto multiplicado por la columna “margen2” de la tabla Fabricante, en caso contrario el precio de venta se calcula con el precio de venta del producto multiplicado por la columna “margen1” de la tabla Fabricante.

Si el producto no existiese devuelve -1.
1,5 ptos

4. Utilizando la función anterior realiza una carga masiva sobre la tabla stock, la columna cantidad tomará el valor 3 en todos los registros.
   0,25 ptos
   Nota: Carga masiva= Insert sobre una tabla cuyos registros son tomados desde un select.

5. Crea la **tabla VENTA** con las siguientes columnas :
   IdProducto: Entero(10), clave primaria compuesta y foránea la columna código de la tabla Producto.
   Fecha:TimeStamp, clave primaria compuesta, Fecha en la que se realiza la venta.
   Cantidad: Entero. Cantidad de la venta de ese producto.
   PVP_total: Precio total de la venta.
   0,5 ptos

6. Crea el **disparador insertVenta** asociado a la tabla VENTA. Cada vez que se inserte un registro en la tabla VENTA se descontará sobre la tabla Stock la cantidad que hay de ese producto. Es decir Stock.cantidad=Stock.cantidad-VENTA.cantidad.
   1 ptos

7. Crea el **procedimiento venta(nombre, cantidad)**, recibe como parámetro el nombre del producto VARCHAR(40) y la cantidad INT que se desea comprar.
   Este procedimiento insertará en la tabla VENTA un registro siempre y cuando haya stock suficiente. Es decir stock.cantidad >=cantidad.
   1 ptos

8. Repite el **procedimiento anterio**r para asegurarte que no haya inyección SQL.
   1 ptos

9. Crea la tabla **pedidoProveedor**, con los campos
   idProducto: Integer(10) primary key y foránea de la clave primaria de Productos.
   Fecha: TimeStamp primary Key. Es la fecha en la que se realiza el pedido.
   Cantidad: Integer. Es la cantidad que se hace de dicho pedido al proveedor.
   Pendiente: Lógico. Si true indica que ese pedido está pendiente de ser servido.

   Inserta varios pedidos a proveedor.
   0,5 ptos

10. crea un disparador sobre la tabla PedidoProveedor asociado al evento Update.
    Este disparador solo funciona cuando cambia el estado de la columna “Pendiente” de true a false. Cuando esto suceda se actualizará sobre la tabla Stock la columna cantidad, sumándose a esa cantidad la cantidad del producto servida.
    1,5 ptos

11. Realiza el procedimiento Servir(nombreFabricante)
    Recibe como parámetro el nombre de un Fabricante VARCHAR(40)

Este procedimiento sirve los pedidos pendientes de un fabricante. Para ello pone a false el campo pendiente de la tabla PedidoProveedor de todos los pedidos pendientes de ese fabricante.
RECUERDA QUE LA INSTRUCCIÓN UPDATE LLEVA UN PREDICADO WHERE, ESTE PUEDE SER UNA SUBCONSULTA SI UTILIZA LOS OPEDORES IN, NOT IN, EXITS….

1,5 ptos